APPLE LISA
FLOATING POINT
SAVAGE BENCHMARK 2

October 2, 1987

Written by

David T. Craig

736 Edgewater
Wichita, KS 67230

Table of Contents

Introduction 2
Benchmark Results 3
Pascal Source Code Listing 6
Assembly Source Code Listing 9
WorkShop Compilation Listings 18

Introduction

This document describes the testing of the Savage benchmark upon the Apple Lisa computer. The Savage benchmark tests the numeric performance of computers. Concentration is paid to the calculation of the following mathematical functions:

Tangent
ArcTangent
Exponent (e^x [e = 2.71...])
Logarithm (base e)
Square Root

The benchmark was run on a Lisa 2 system which consisted of 1 MByte of RAM, an internal 10 MByte hard disk drive, and an internal micro disk drive. The test program was written in Pascal and tested in the Lisa WorkShop 3.9 development environment.

The Lisa computer uses a floating point engine called SANE (Standard Apple Numeric Environment) which was designed with numeric accuracy as the most important criteria. For a comprehensive discussion of SANE (i.e., what it is and why it is better) refer to the Apple Computer publication titled "Standard Apple Numeric Environment" (the Lisa WorkShop 3.0 manual set [product # 620-6149-B] also provides a very detailed discussion of SANE, but has little information concerning why SANE is better than other numeric engines).

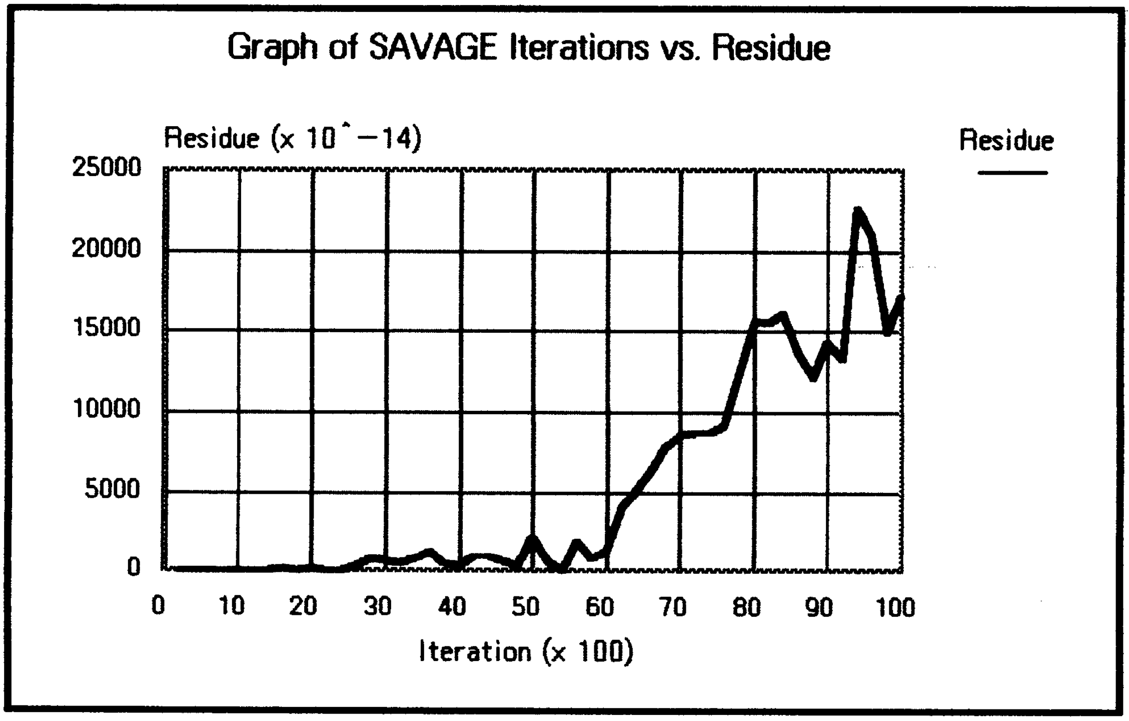
Benchmark Results

The following listing was directly generated by the Savage Residue benchmark program. The listing shows the number of iterations performed, the computation time (in seconds), and the computation residual value (the magnitude of the difference between the true value and the calculated value).

```
Apple Lisa Floating Point SAVAGE RESIDUE Benchmark

Savage Benchmark is executing ...

Iterations     Duration (sec)     Residue
----------     --------------     ------------------
       200             19.020     0.00000000000002
       400             38.239     0.00000000000001
       600             57.636     0.00000000000017
       800             76.863     0.00000000000035
      1000             96.474     0.00000000000031
      1200            115.780     0.00000000000030
      1400            134.929     0.00000000000110
      1600            154.393     0.00000000000164
      1800            174.032     0.00000000000136
      2000            193.782     0.00000000000202
      2200            213.294     0.00000000000093
      2400            232.565     0.00000000000114
      2600            251.774     0.00000000000302
      2800            270.953     0.00000000000808
      3000            290.407     0.00000000000721
      3200            310.041     0.00000000000477
      3400            329.747     0.00000000000744
      3600            349.493     0.00000000001306
      3800            369.325     0.00000000000515
      4000            389.197     0.00000000000394
      4200            408.981     0.00000000001003
      4400            428.393     0.00000000000908
      4600            447.763     0.00000000000710
      4800            467.092     0.00000000000337
      5000            486.364     0.00000000002172
      5200            505.694     0.00000000000614
      5400            524.936     0.00000000000057
      5600            544.273     0.00000000001869
      5800            563.537     0.00000000000868
      6000            583.291     0.00000000001282
      6200            603.065     0.00000000003960
      6400            622.800     0.00000000004946
      6600            642.631     0.00000000006289
      6800          662.393       0.00000000007725
      7000          682.252       0.00000000008491
      7200          702.079       0.00000000008717
      7400          721.968       0.00000000008664
      7600          741.870       0.00000000009070
      7800          761.843       0.00000000012604
      8000          781.807       0.00000000015592
      8200          801.987       0.00000000015477
      8400          821.613       0.00000000016094
      8600          841.129       0.00000000013611
      8800          860.606       0.00000000012138
      9000          880.060       0.00000000014387
      9200          899.493       0.00000000013334
      9400          918.896       0.00000000022605
      9600          938.312       0.00000000020860
      9800          957.717       0.00000000014886
     10000          977.110       0.00000000017072

  That's all, Folks...
```

Apple Lisa Floating Point SAVAGE RESIDUE Benchmark (2-Oct-87)

In order to better understand the pattern behind the residues, the iteration count vs. the residue values was plotted. From the below plot one can see that the residues generally increase as the iteration count increases. But the increase is erratic. For example, iteration count 9400 had a higher residue than 9200, and iteration count 9600 was smaller than the residue of iteration count 9400.



Pascal Source Code Listing

```
{ [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
  []                                                                  []
  []              FLOATING POINT SAVAGE RESIDUE BENCHMARK             []
  []               FOR THE APPLE LISA AND SANE FP ENGINE              []
  []                                                                  []
  []                          October 2, 1987                         []
  []                                                                  []
  []       Compiled using Lisa Pascal 3.76 in the Lisa WorkShop       []
  []                                                                  []
  []                         by David T. Craig                        []
  []                 [736 Edgewater, Wichita, KS 67230]               []
  []                                                                  []
  []     Part of source code listing from Byte, Oct. 1987, p. 277     []
  []                                                                  []
  [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][] }

PROGRAM Savage_Residue_FP_Benchmark;

  {$R-  } { Turn OFF range checking }
  {$ASM+} { Turn ON  68000 assembly code listing }

  USES {$L-}
       {$U LISA/HARDWARE.OBJ} HARDWARE, { Lisa Hardware  interface }
       {$U LISA/SANELIB.OBJ } SANE;     { Lisa FP engine interface }
       {$L+}

  TYPE
    t_FP_Number = Extended; { 80-bit precision FP value (Apple SANE type) }
                            { accurate to 19 decimal places               }

  VAR
    g_iteration : INTEGER;      { Iteration counter      }
    g_time      : MilliSeconds; { Execution time (in ms) }
    g_residue   : t_FP_Number;  { Residue value          }

{ ===================================================================== }

  PROCEDURE Show_Execution_Results(show_title_line : BOOLEAN;
                                   iteration_count : INTEGER;
                                   exec_duration   : MilliSeconds;
                                   residue_value   : t_FP_Number);

    BEGIN { ----- Show_Execution_Results ----- }
      IF show_title_line THEN
        BEGIN
          WRITELN('Iterations    Duration (sec)    Residue');
          WRITELN('----------    --------------    ------------------');
        END
      ELSE
```

```
        WRITELN(iteration_count:10,'    ',
               (exec_duration / 1000.0):14:3,'    ',
               ABS(residue_value):18:14);
    END;  { ----- Show_Execution_Results ----- }

{ ====================================================================== }

  PROCEDURE Perform_SAVAGE_Algorithm(    iter_loop : INTEGER;
                                     VAR exec_time : MilliSeconds;
                                     VAR residue   : t_FP_Number);

    VAR
      i      : INTEGER;      { Loop counter          }
      a      : t_FP_Number;  { Computation variable   }
      start  : MilliSeconds; { Starting  time (in ms) }
      finish : MilliSeconds; { Ending    time (in ms) }

    BEGIN { ----- Perform_SAVAGE_Algorithm ----- }
      start := Timer; { Start the timing }

      a := 1.0; { Initialize the benchmark variable }

      FOR i := 1 TO iter_loop DO { Perform the actual benchmark }
        a := Tan(ArcTan(Exp(Ln(Sqrt(a*a))))) + 1.0;
             { <---------- = a ----------> }

      finish := Timer; { End the timing }

      exec_time := finish - start; { Determine real execution time }

      residue := ((iter_loop * 1.0) + 1.0) - a; { Return the residue value }
    END;  { ----- Perform_SAVAGE_Algorithm ----- }

{ ====================================================================== }

  BEGIN { ----- Savage_Residue_FP_Benchmark ----- }
    WRITELN('Apple Lisa Floating Point SAVAGE RESIDUE Benchmark');
    WRITELN;
    WRITELN('Savage Benchmark is executing ...');
    WRITELN;

    { Make certain the SANE FP engine code is loaded }

    g_residue := SIN(10.0) * 1.2345; { Don't Care values }

    { Perform the actual SAVAGE benchmarks }

    Show_Execution_Results(TRUE,0,0,0.0);

    FOR g_iteration := 1 TO 50 DO
      BEGIN
        Perform_SAVAGE_Algorithm       (g_iteration*200,g_time,g_residue);
        Show_Execution_Results   (FALSE,g_iteration*200,g_time,g_residue);
      END;

    { Aufwiedersehen ... }

    WRITELN;
    WRITELN('That''s all, Folks...');
  END.  { ----- Savage_Residue_FP_Benchmark ----- }

{ [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
  []                                                                  []
  []                            F I N I S                             []
  []                                                                  []
  [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][] }
```